Cloning the repo and working with Git

MelvMay-GG edited this page on Mar 1, 2013 · 3 revisions

Git is a powerful system for collaborating on projects and is something that you NEED to master if you intend to participate in and keep up to date with the Torque2D MIT (T2DMIT) project.

The purpose of this guide is to get you started with T2D MIT and will focus exclusively on the use of Git-Bash.

- This guide will NOT go over all the possible uses of each Git command.
- This guide will NOT cover the many gui frontends that are available for Git (SmartGit, Git for Windows, etc.).

Going through this tutorial at least once using Git-Bash will provide you with a strong grasp on all the concepts required to work with git repositories.

You may then feel free to use the gui frontend of your choice but taking the time to learn the basics will pay off greatly down the line.

Important Git concepts

Here are the basic terms you should familiarize yourself with before embarking on your journey.

- Repository / Repo : This is the project's source code that resides on github.com's servers. You cannot modify the contents of this repository directly unless you were the one who created it in the first place.
- Fork : *Forking* a project will create a copy of the original repository that you can modify as you please. *Forked* projects will appear in your own github.com account.
- Cloning : this will clone an online repository to your hard drive so you may begin working on your modifications. This local copy is called your local repository.
- Branch : A *branch* is a different version of the same project. In the case of T2DMIT, you will see 2 branches : the master branch and the development branch.

Pages 56

Find a Page...

Home
Android Development Guide
Animation Guide
AnimationAsset Guide
Asset Manager Guide
AssetBase Guide
Audio Guide
AudioAsset Guide
Batch Rendering Guide
Behavior Connections
Behavior Tutorial
Blending Guide
Callback Guide
Changelog
Cloning the repo and working with Git
Show 41 more pages...

Clone this wiki locally

https://github.com/GarageGames/Torque2D.wiki.git

Clone in Desktop

Remote : A ***remote*** is simply an alias pointing to an online ***repository***. It is much easier to work with such aliases than typing in the complete URL of online ***repositories*** every single time.

- Staging Area : Whenever you want to update your online ***repository*** (the one appearing in your github.com account), you first need to add your changes to your ***staging area***. Modifying files locally will not automatically update your ***staging area***'s contents.

Important Git commands

- Fetch : ***git fetch*** will download the current state (containing updated and newly created ***branches***) of an online ***repository*** without modifying your local repository. It places its results in .git/FETCH_HEAD.
- Merge : ***git merge*** will merge the modifications of another ***branch*** into the current working ***branch***.
- Pull : ***git pull*** is actually a combination of ***git fetch*** and ***git merge***. It fetches the information from an online ***repository***'s ***branch*** and ***merges*** it with your local copy.
- Add : Whenever you modify a file in your local ***repository*** or create a new file, that file will appear as unstaged. Calling ***git add*** allows you to specify files to be added to your ***staging area***.
- Commit : A ***commit*** records a snapshot of your ***staging area***, making it ready to be ***pushed*** to an online ***repository***.
- Push : ***git push*** will take all of your locally committed changes and upload them to a remote repository's ***branch***.

Don't worry if these words aren't familiar to you just yet, going through the actual process will (hopefully!) make everything clear.

Step by Step : From cloning the repo to pushing your first changes

1- Basic set up and forking

First up, you should create an account on Github.com and download git for your preferred Operating System

http://git-scm.com/downloads

Next, head on over to GarageGames' Torque2d repository :

https://github.com/GarageGames/Torque2D

In the top-right corner, you will see a 'Fork' button, click it!

Github will now create a copy of the T2DMIT project in your account and will give you read and write access to this copy.

Note that your forked copy will not automatically update itself as the T2DMIT project gets updated : your forked copy represents the T2DMIT project as it existed at the moment when you have forked it.

To stay up to date with the latest T2DMIT repository changes, be sure to Watch and Star the repository from its github.com page. All updates will then appear in your personal news feed when you visit github.com

![github's Watch and Star] (http://www.s01l.com/gitbuttons.png "Watch and Star a repo")

Congratulations! You are now ready to set up your local environment

Start up Git Bash.

type in

```
git config --global user.name "Your Name here"
```

This will be the name that shows up in the ***commits*** whenever you make a modification

type in

```
git config --global user.email "your_email@domainname.com"
```

Make sure to enter the email address that is associated with your github account!

2- Cloning the repo

Now that our username and email are configured, it is time to ***clone*** the ***repository***.

When you login to github.com on the web, you should see your repositories on the right-hand side of the screen. Navigate to your_user_name/Torque2D.

You should now see the files contained in your repository and the time since each file was last modified in the currently selected branch (Remember that this reflects the online repository, not your local changes!)

Near the top of the screen, you will see the following

![github's repo URL] (http://www.s01l.com/gitURL.png "Github")

Copy this address to your clipboard and return to gitbash.

By default, the command "git clone repository_name" will replicate the directory structure of your forked repository wherever the default Git Config tells it to.

If you want to clone the repository somewhere else, simply add the pathname at the end of the command

```
git clone repository_name /path/to/localrepo/
```

- If you've ever used DOS or DOSBOX, you might be used to typing C: to access a physical drive. Take note that GitBash works like a Linux command-line (Bash). In this system, accessing the C drive would be achieved by typing in cd /c/.
- To list the contents of a directory or folder, old-school PC users would type in dir, in this environment, use ls instead.
- Finally, to PASTE the contents of your clipboard into Git Bash, Ctrl+V won't work. Use Shift+Insert instead.

Now type in the following line to clone your forked repository into directory dev/T2DMIT on your E: drive

```
git clone git@github.com:your_username/Torque2D.git /e/dev/T2DMIT
```

You will see that GitBash will display the cloning's progress as it downloads the files. Once complete, head over to your chosen directory where you should see the files!

3- Setting up Remotes

When you first clone a repository, git will automatically create a remote named 'origin' for you,.

Now remember that this remote will point to YOUR repository, which was forked from the official T2D MIT repository. As mentioned before, over the course of development, the T2D MIT repository will be updated but YOUR repository will not.

To remedy this, we want to setup a remote that will track the original repo (T2DMIT).

Using Git Bash, navigate to your project's directory. Once you are in the right place, you should see (master) appear after the current directory's name. This represents the currently active branch.

EVERY ACTION YOU TAKE WILL BE APPLIED TO THE CURRENTLY ACTIVE BRANCH.

Return to Git Bash and type in

```
git remote add upstream https://github.com/GarageGames/Torque2D.git
```

this will add a new remote called 'upstream' which points to the official Torque2D repository.

If you want to see where each remote is pointing, type in :

```
git remote -v
```

4 - Managing Branches

Making changes directly to your master branch is a bad idea. You should always have a working branch to try out your modifications on.

To list the available branches for your current project, type in :

```
git branch
```

To create a new branch, naming it whatever you want, type in :

```
git branch branch_name
```

To delete a branch, type in :

```
git branch -D branch_name
```

To switch to a branch, making it the currently active branch, type in :

```
git checkout branch_name
```

As an example, in order to return to your master branch, you would type in

```
git checkout master
```

4a - T2D Branches : master & development

There are two main branches to the Torque 2d MIT project.

- The Master branch is the 'official' one, the stable version that you should use if you want to create a game using the technology.
- The Development branch is updated much more often, and will feature works in progress, new features, new toys and various tweaks to the engine. Once changes in the development branch are deemed final and tested properly, these changes will be pushed to the master branch.

When you first clone the repository, you will find yourself in the master branch.

I suggest creating a development branch by typing the following series of commands :

```
git branch development
git checkout development
git pull upstream development
```

If you've been reading from the beginning, this should be fairly simple to understand.

- First, we have created a new local branch, called development
- Secondly, we switch to the development branch with the checkout command, making 'development' our currently active branch
- Finally, we target the most recent version of the repository that is linked to our upstream remote(GarageGames' T2D repository) and pull in the contents of its development branch into our currently active local branch.

Everytime you see new changes on the official Torque 2d MIT development branch, just navigate to your local repository and follow steps 2 and 3. Remember to Watch and Star the official repository!

At this point, you are ready to start working with T2DMIT!

5 - Final Step : Adding files, Committing changes and Pushing

Whenever you want to update your online repository with your local changes, you need to follow these basic steps :

- git add all the new or modified files to your staging area
- Execute a git commit to take a snapshot of your local staging area
- git push the snapshot to your online repository

Whether you have modified an existing file or created a new file in the local repository, the procedure remains the same.

Simply type in

```
git add modified_filename1 modified_filename2 newly_created_filename1
```

This will add the specified files to your staging area.

If you do not call ***git add*** on a modified file, the staging area will simply keep the version of the file which existed when it was last added via ***git add***.

You may also call

```
git add .
```

to add all modified or new files in your entire project to your staging area.

Once your staging area is ready, you must commit your changes by typing

```
git commit -m 'hopefully relevant message about this commit'
```

If you run git status or look at your project via github.com, you will see the messages of each commit next to the modified files as well as the date of the last commit.

Take a look at GarageGames' official T2DMIT repository's development branch for good examples of commit messages.

Finally, you push your commit to your online repository by typing :

```
git push remote_name branch_name
```

or more specifically :

```
git push origin development
```

The above command will push the changes from your currently active branch to your online repository's development branch. You cannot push your commit to the upstream remote as you do not have write access to GarageGames' repository.

Once you are confident that your changes should be part of the Torque2D engine's main repository, you should head on over to the following post on GarageGames' forums to learn how to submit a pull request

http://garagegames.com/community/blogs/view/22166

Simply scroll down to the section called : How do I participate in growing Torque 2D?

Work methods

While writing this guide, I've noticed that Git usage will vary wildly based on your experience level and personal preferences. As long as you understand the logic of your actions, there is no 'wrong' way to go about using Git.

When you work on code on your local machine, I strongly suggest doing so in a branch that is separate from your master and development branches.

The master and development branches should reflect T2DMIT's latest changes.

You can choose to push your work branch to your online repository or you can simply merge your work branch's changes into your own development branch. Experiment and find the way which suits you best.

You may also obtain a zipped file of the project's most recent state from the project's github.com page. While this method seems simpler, the git method will allow you much more flexibility.

Recommended reading

The Git Reference docs explain everything about Git with really clear examples : http://gitref.org/index.html

I strongly suggest bookmarking http://www.kernel.org/pub//software/scm/git/docs/ for the most direct and complete information on each command.

You can also run through this amazing and simple tutorial. It looks deceptively simple but it teaches you everything you need to know about using Git : http://rogerdudler.github.com/git-guide/

You can also download the 'Git cheatsheet' and keep it handy : http://rogerdudler.github.com/git-guide/files/git_cheat_sheet.pdf